Grand Unified File Index

Development, Deployment, and Performance Update

Dominic Manno

May 22, 2019

NNSA
National Nuclear Security Administration

Managed by Triad National Security, LLC for the U.S. Department of Energy's NNSA

LA-UR-19-24645

Acknowledgments

- Some slides and content/diagrams provided by LANL colleagues: David Bonnie, Gary Grider, Jason Lee, Brad Settlemyer

Agenda

- HPC at LANL
- GUFI Overview
- Development Update
- Deployment Strategies
- Performance Details
- What’s next?

LANL’s HPC Environment

HPC at LANL

- Eight decades of weapons computing support to keep the nation safe
  - Simulation to determine stability, defects, etc.
- Cutting edge technology enables large, long-running, multi physics 3D simulations
  - Jobs can last months running on 80% of the machine

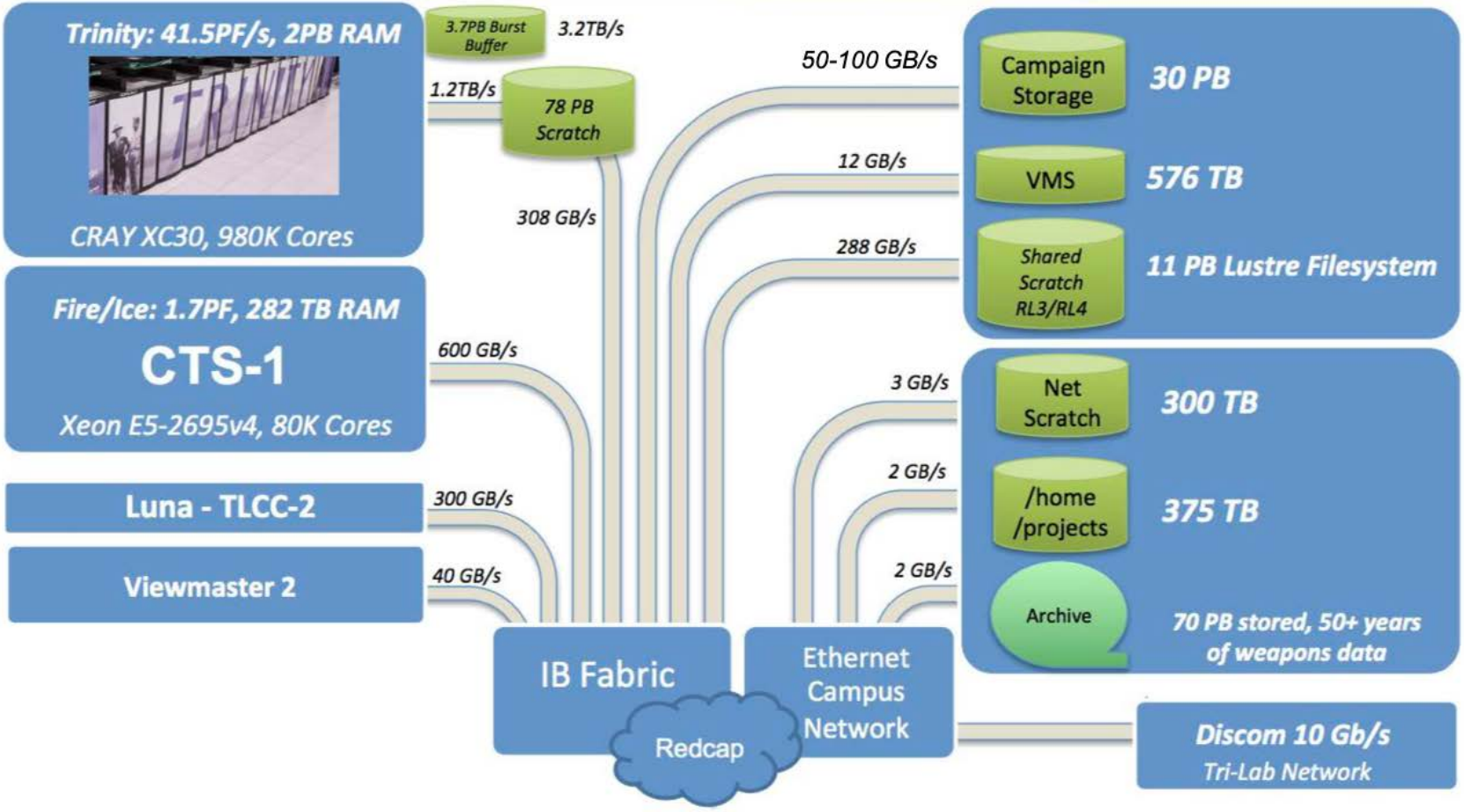
Better Science Calls for Better Computers

Roadrunner (2007)
1st Petaflop/Accelerator Platform

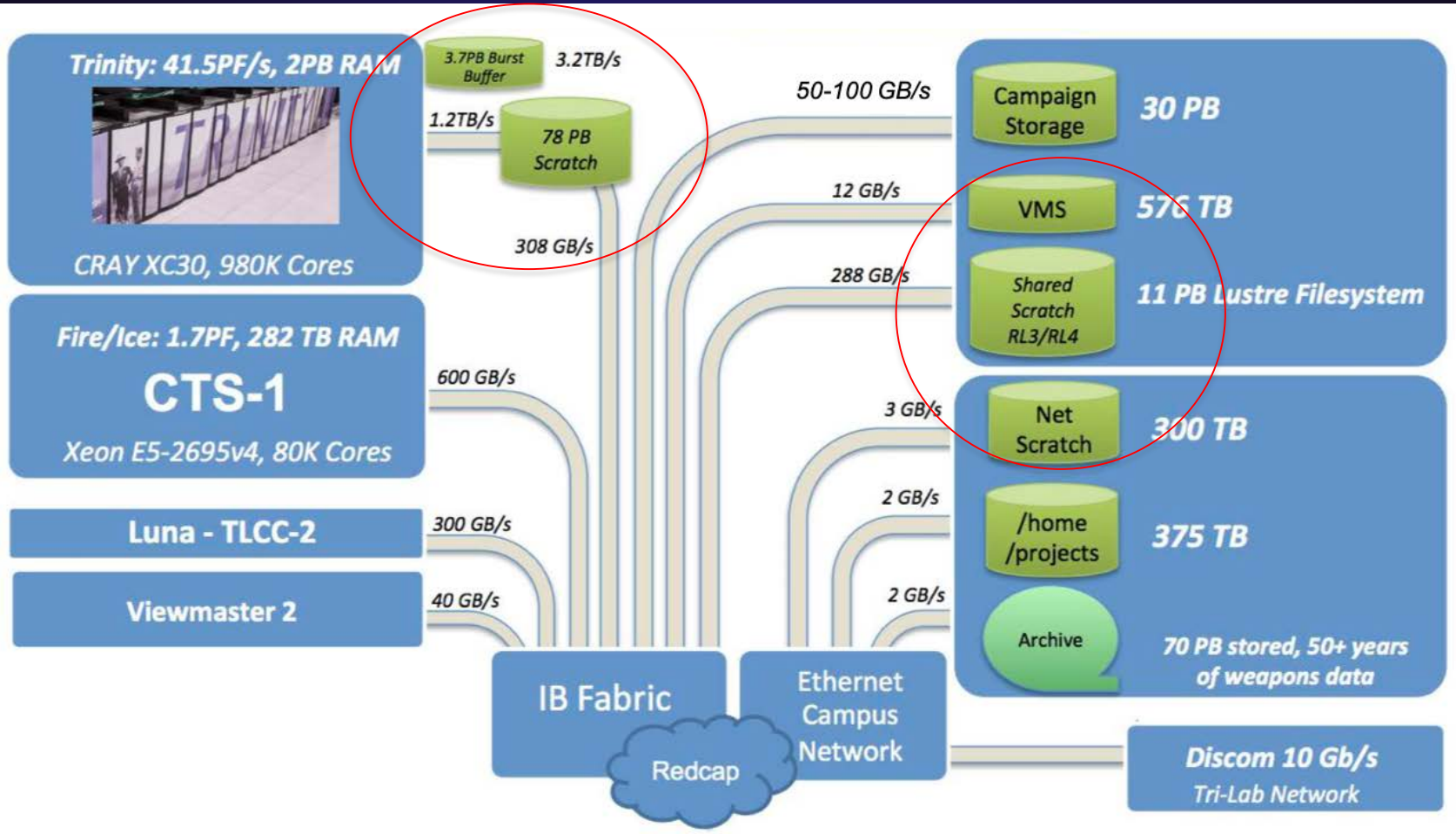
Cielo (2011)
1.7 Petaflop Platform

Trinity (2015)
~20 Petaflops, 4 PB Burst Buffer

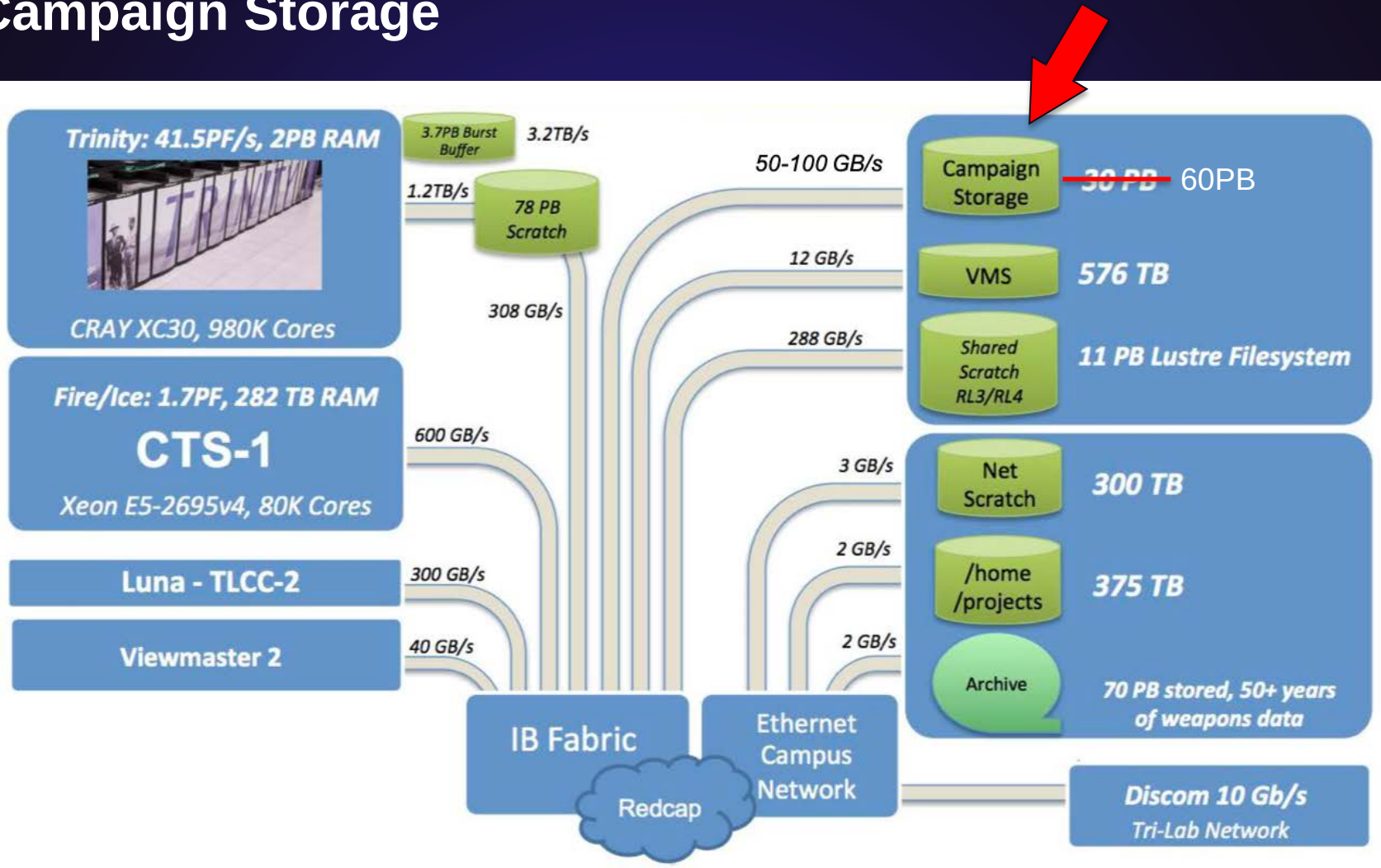
Storage Tiers



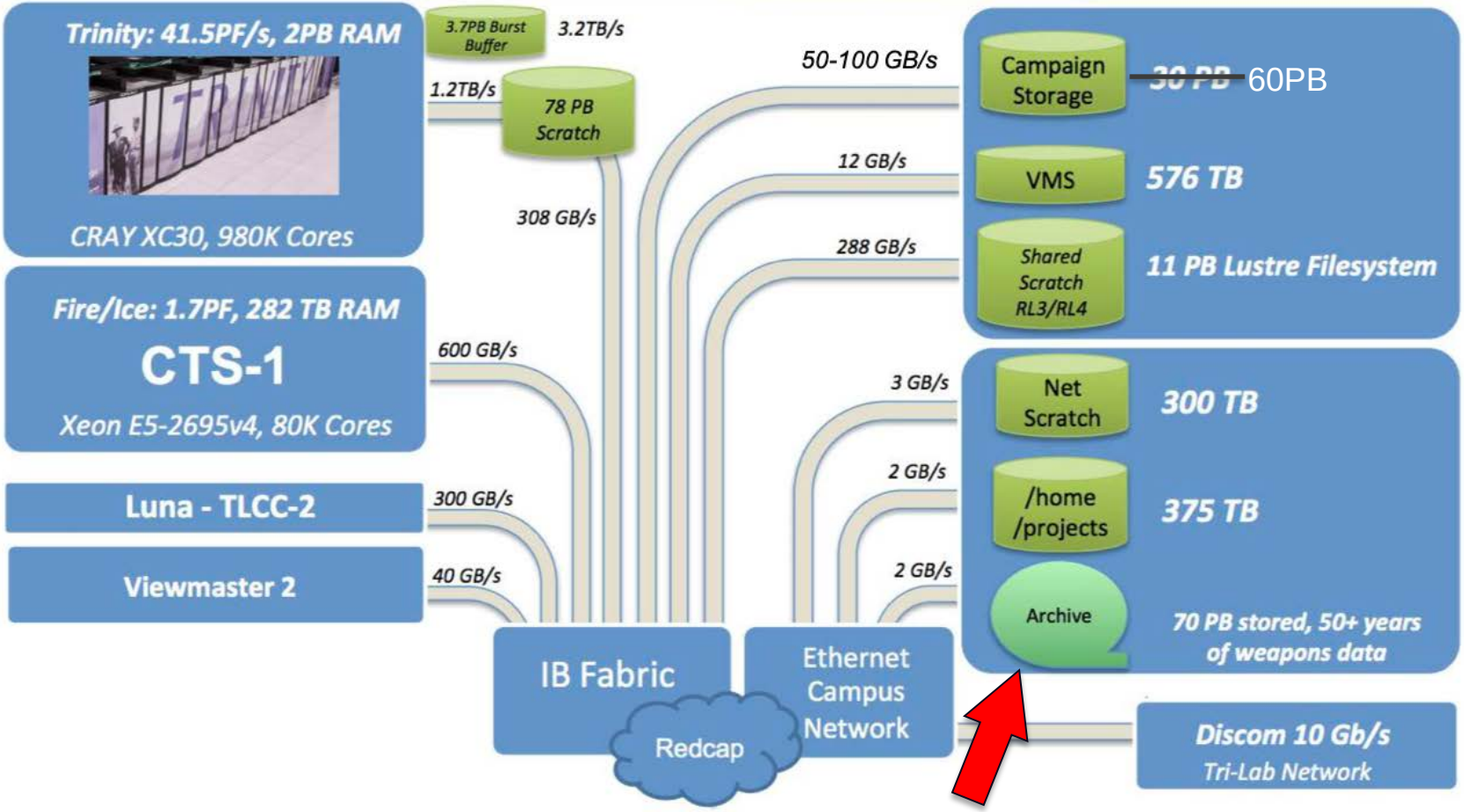
Scratch – lustre (mostly)



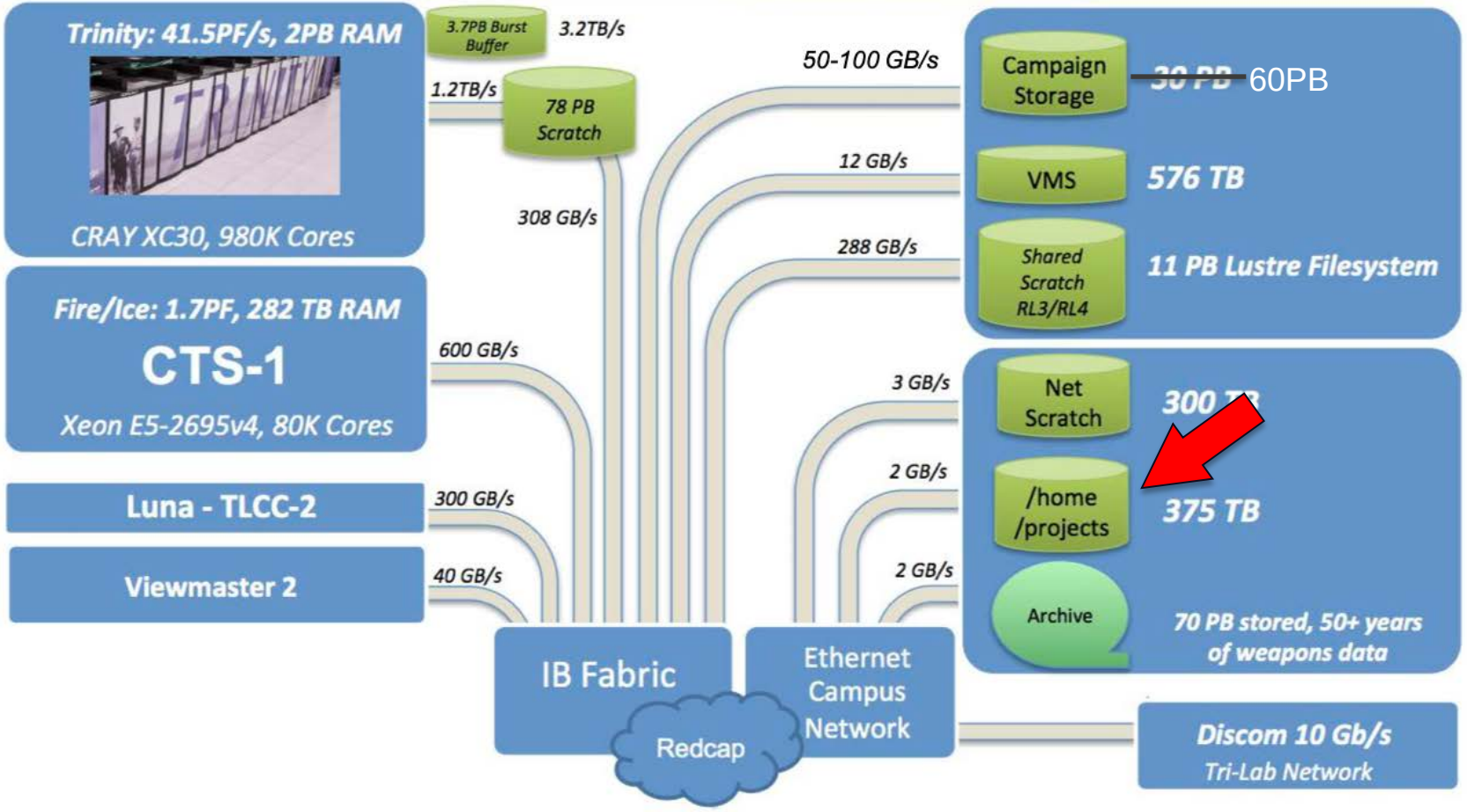
Campaign Storage



Archive


Trinity: 41.5PF/s, 2PB RAM
CRAY XC30, 980K Cores
Fire/Ice: 1.7PF, 282 TB RAM
CTS-1
Xeon E5-2695v4, 80K Cores
Luna - TLCC-2
Viewmaster 2
3.7PB Burst Buffer
3.2TB/s
1.2TB/s
78 PB Scratch
308 GB/s
600 GB/s
300 GB/s
40 GB/s
50-100 GB/s
12 GB/s
288 GB/s
3 GB/s
2 GB/s
2 GB/s
IB Fabric
Redcap
Ethernet Campus Network
Campaign Storage
30 PB
60PB
VMS
576 TB
Shared Scratch RL3/RL4
11 PB Lustre Filesystem
Net Scratch
300 TB
/home /projects
375 TB
Archive
70 PB stored, 50+ years of weapons data
Discom 10 Gb/s
Tri-Lab Network

Oh yeah – and home/projects



Metadata problem?

- This model depends on users knowing about their data
  - Where did it get written?
  - Does it need to be backed up? If so, did I already save a copy?
  - Good naming and hierarchy
- Without explicit management the archive would collect far too much data
- Need to provide better tools

GUFI Overview

Early Discussions

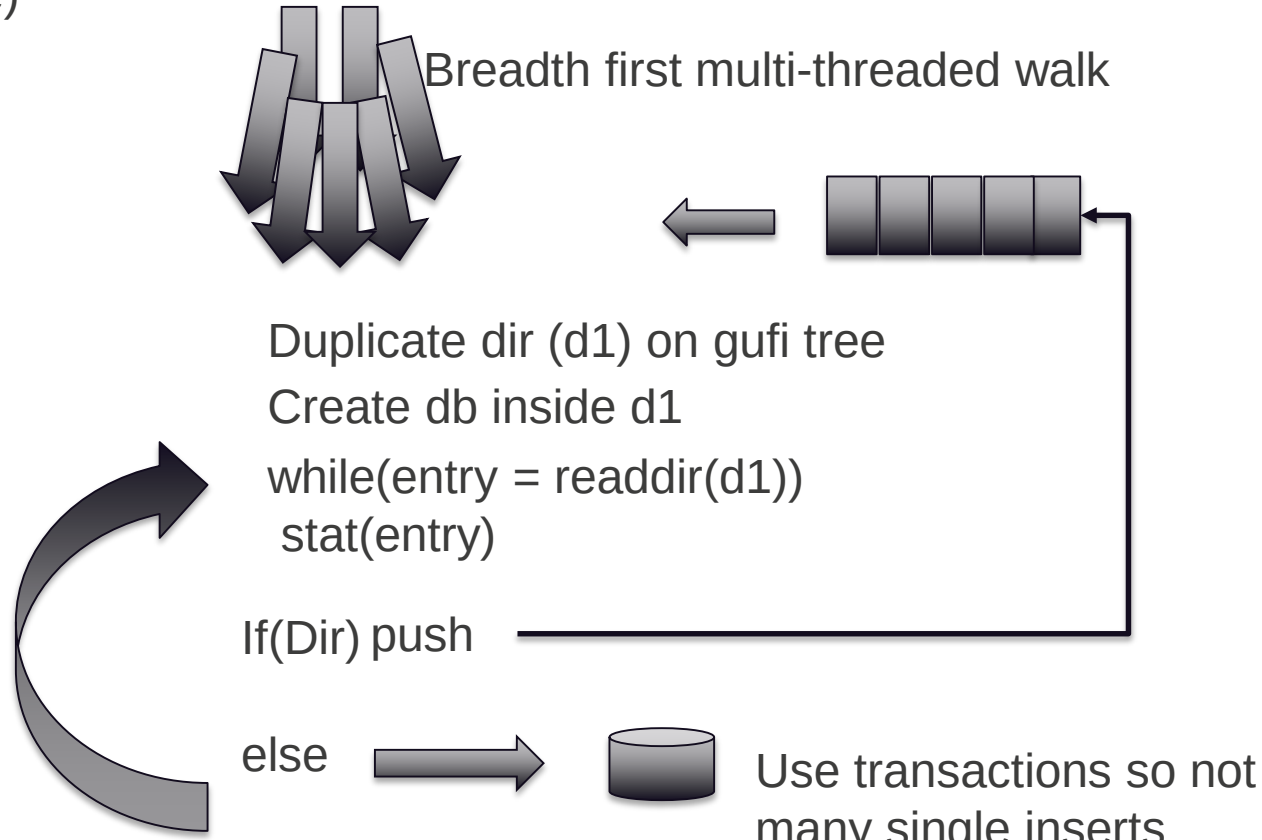
- Provide an index over all tiers of storage
- Securely allow admins (easy) and users to share the index and tools
- Reasonable update times – may need incremental, keep stress on source FS low if possible
- Parallel is key -- threads
- Include xattrs
- Leverage existing technology
- Keep it simple

GUFI Design

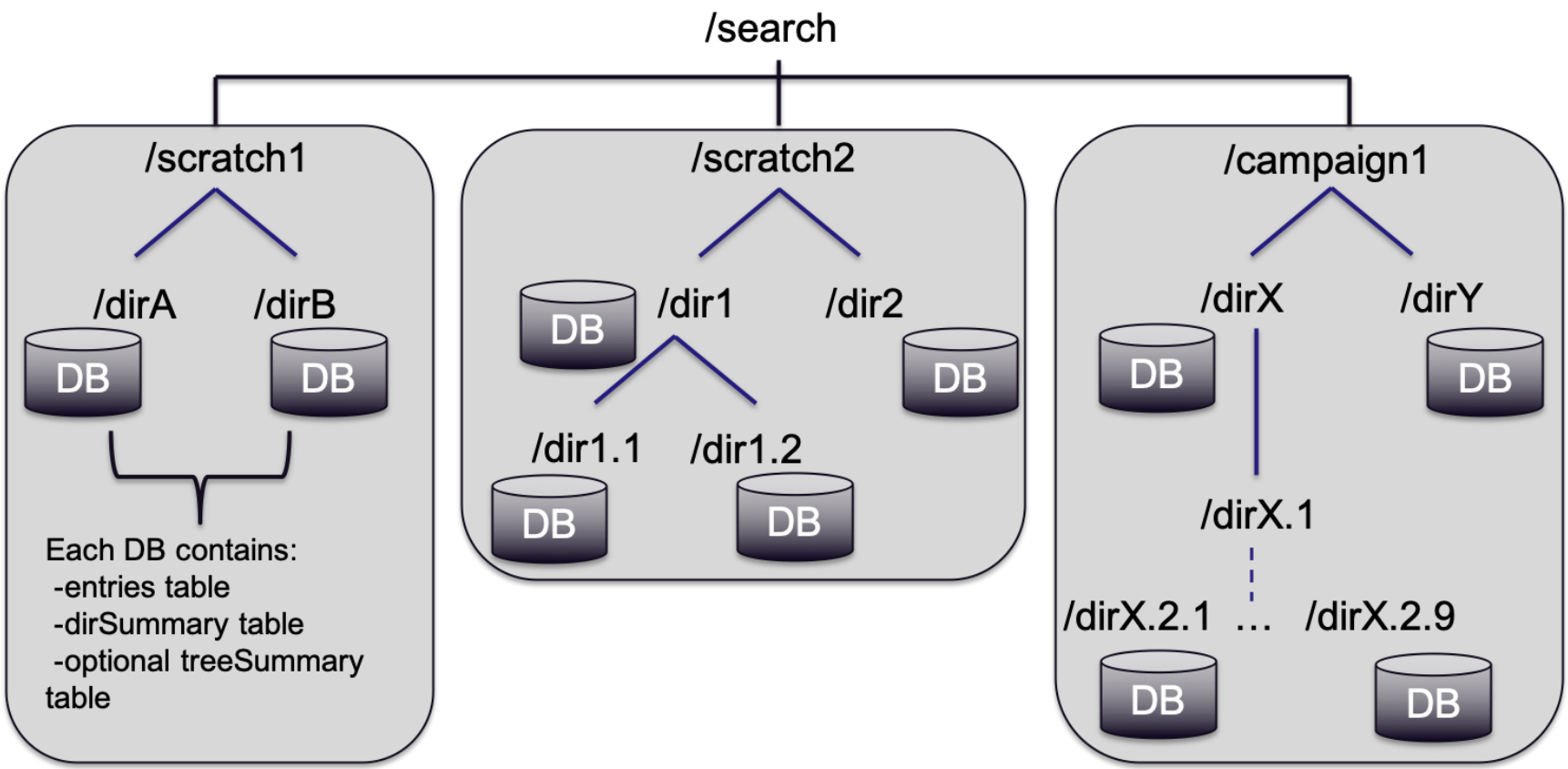
- Re-create source FS tree
  - Maintain ownership and permissions on the newly created tree
  - Secure – we already depend on these permissions on the source
- Use embedded DB in every dir
  - sqlite
  - This is where all file information goes
- Threads!

GUFI Design – over simplification of ingest

- Assume building GUFI index as walking source tree (bfwi w/ full build index mode)



GUFI Design


/search
/scratch1
/dirA
/dirB
DB
DB
Each DB contains:
-entries table
-dirSummary table
-optional treeSummary table
/scratch2
DB
/dir1
/dir2
DB
/dir1.1
/dir1.2
DB
DB
/campaign1
/dirX
/dirY
DB
DB
/dirX.1
/dirX.2.1
…
/dirX.2.9
DB
DB

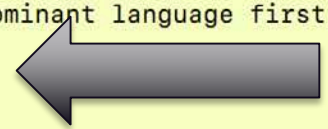
GUFI Design

gufi — -bash — 123×35

```
[bws@pn1809254:gufi:549 [master]]$ sloccount src/
Creating filelist for src
Categorizing files.
Finding a working MD5 command....
Can't exec "md5sum": No such file or directory at /opt/local/bin/break_filelist line 688, <CODE_FILE> line 15.
Found a working MD5 command.
Computing results.


SLOC    Directory       SLOC-by-Language (Sorted)
6541    src             ansic=5739,cpp=802


Totals grouped by language (dominant language first):
ansic:         5739 (87.74%)
cpp:            802 (12.26%)
```



-entries table
-dirSummary table
-optional treeSummary table

/dirX.2.1 … /dirX.2.9

DB DB

Alternative Approaches

- Flatten the namespace
  - Rename on high in the tree is costly
  - Implementing security for users and admins to share is hard and likely a performance hit
- Why not just write MPI or MPI libcircle jobs to do this?
  - Resources
  - Users like find | grep and ls --with-color

Development Update

GUFI_* Tools

- Users are familiar with common tools: find, ls, du, etc.
- Initial user interface is gufi_find and gufi_ls
- Implement as many options as possible using the same flags
  - Create sqlite queries and generate bfq queries based on input
- Don’t write a ton of new code to do this
  - Just wrap existing query tools (bfq) and use the wrapper to generate required queries
  - Python – strings and error handling

Ingest Tools

- File systems provide various interfaces to obtain metadata
- We are implementing and testing some file system specific ingest tools:
  - GPFS
  - Lustre
  - HPSS
- Also testing approach to incremental updates

Hardening

- Build system
  - Incorporate Travis for auto/nightly builds
  - Moved to cmake
  - Verified on RedHat, SUSE, macOS
- Bug fixes

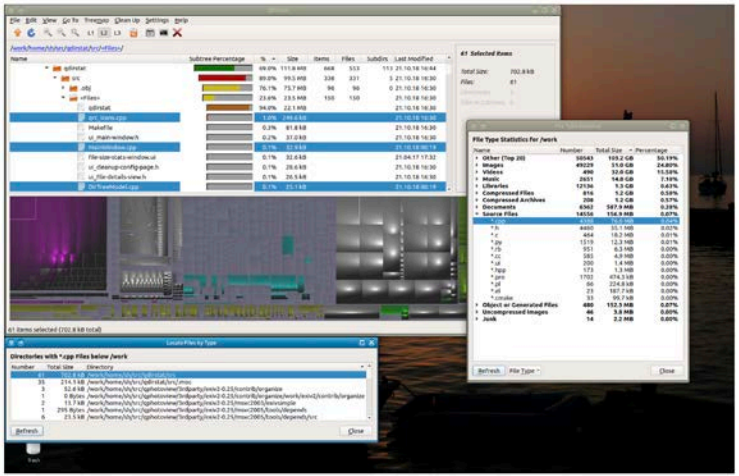
Deployment Strategies

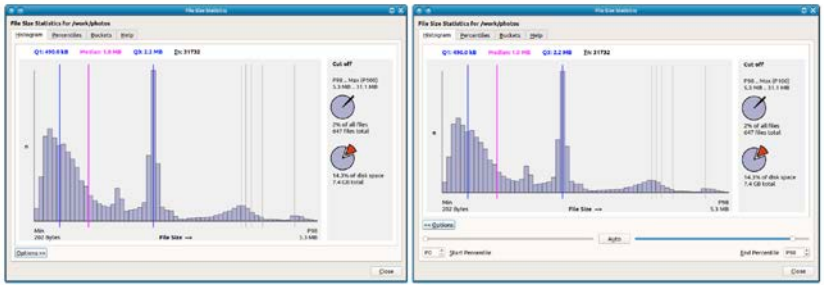
Initial Thoughts on Deployment

- Sqlite queries and current tools ok for admins/power users
  - Don’t expect users to need to know how to write queries
- Normal users want to use find, ls, etc or click to search
  - Read-only fuse can catch calls relied on by find and ls
  - Gufi_find/ls can be used instead
- Frequent interaction means users don’t want to have to hop around to separate servers to get the information they need
- Utilize well-understood methods to allow users to query a remote node
  - SSH (python paramiko)
  - User accounts (passwd, group)
  - Users run as themselves

Reports and Web-interface

- Provide users with an easy to use web interface
- Web-server will run queries based on some user input
- Also present commonly used queries as reports
- Provide a tool to visualize a tree – look for “hot” spots





*images from qdirstat – windirstat linux variant

Performance

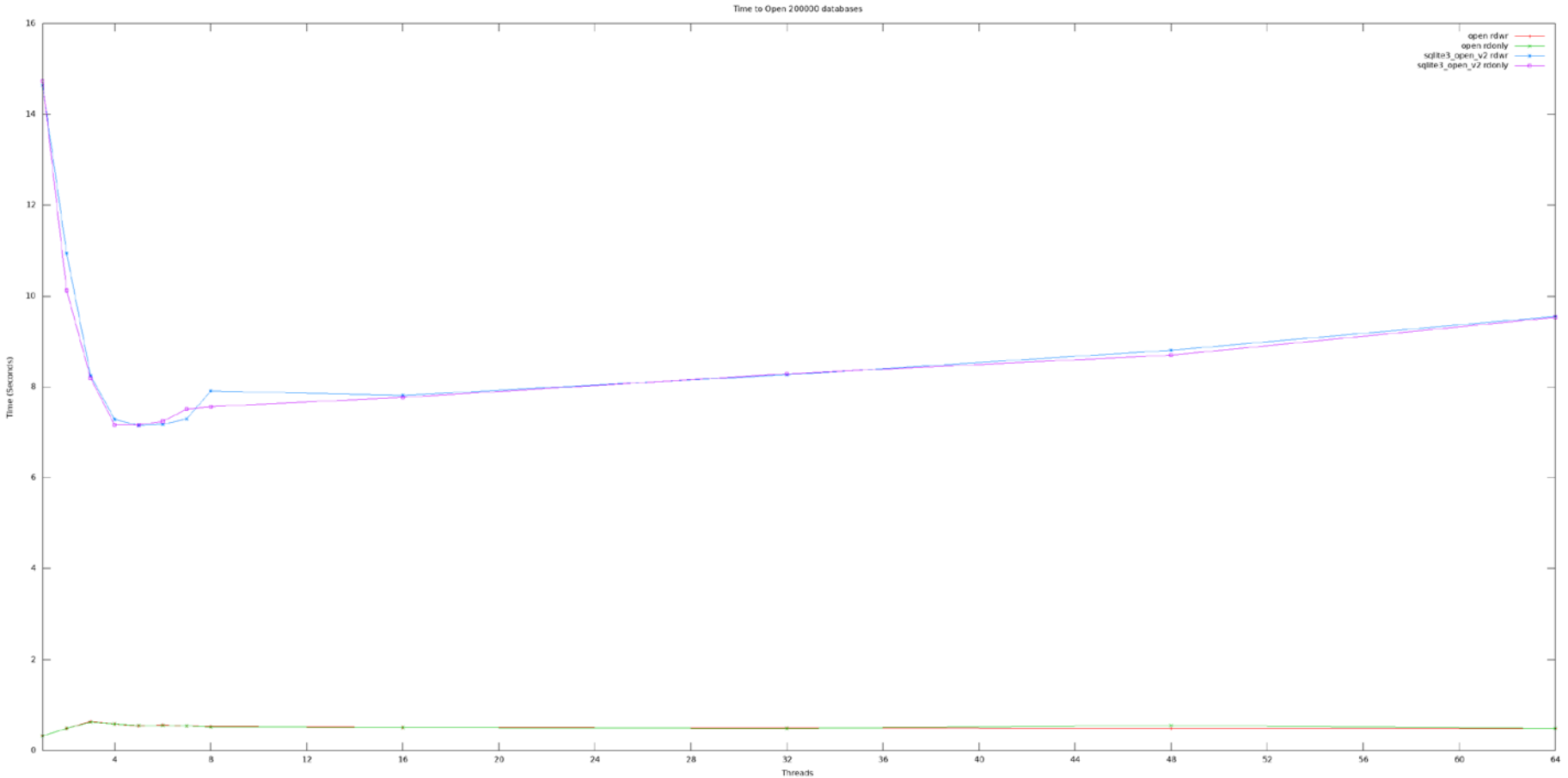
Test Setup

- Single server, Dell R7425
- CPU: AMD Epyc 7401
- Memory: 512 GB
- Kernel 3.10
- Using NVMe SSDs – reported results are only using 1 SSD
- XFS filesystem

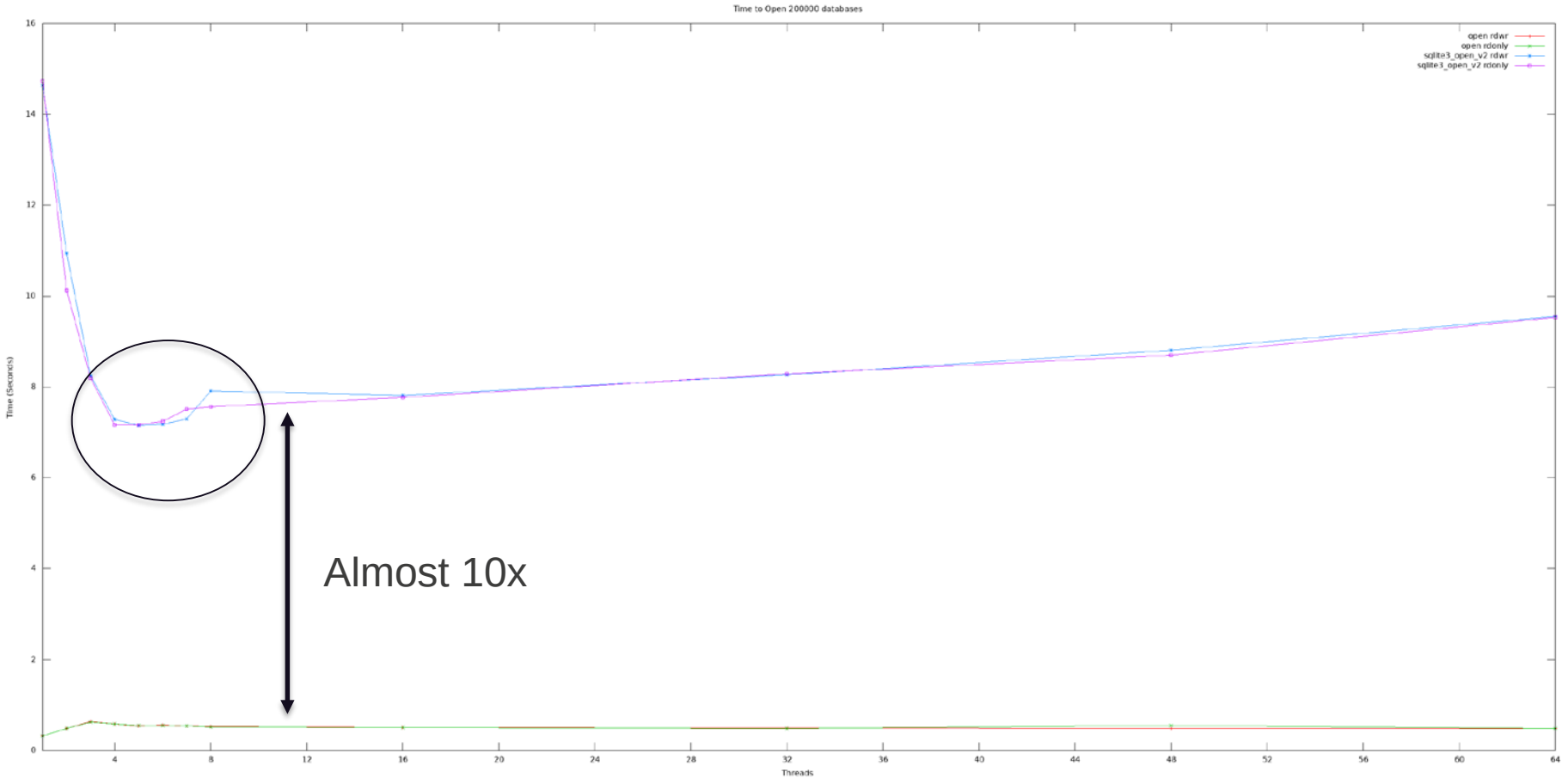
Early Performance From Production Trees

- OK – not what we expected
- Best case ~25x over POSIX
- Worst case only ~4x

Opening DBs Slowing Us Down



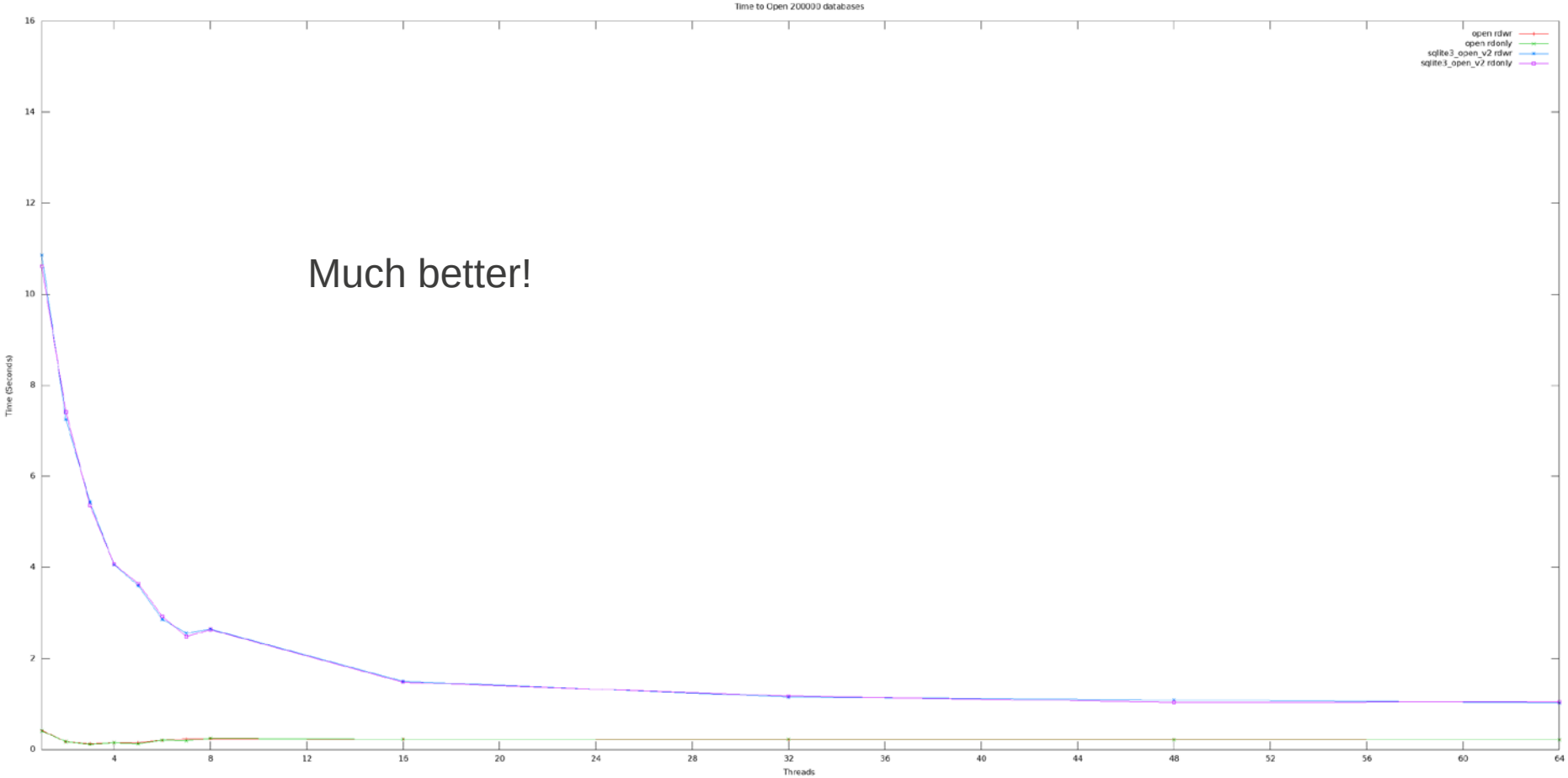
Opening DBs Slowing Us Down



Tuning

- Sqlite3 has protections
- No need for multiple threads to ever access the same DB at the same time
- VFS: unix-none
- Thread-safe = 0

Improved Open Times



Much better!

Improved Query Results

	Find all files in NFS Home as uid 12345		Find all files in NFS Home	
	POSIX	GUFI	POSIX	GUFI
Files	294,188	294,188	13,360,753	13,229,405
Dirs	13,012	13,012	1,633,564	1,622,424
Time	32.1	0.47	2,040	39.2
Files/sec	9,164	625,931	6,549	337,484

Improved Query Results

	Find all files in NFS Home as uid 12345		Find all files in NFS Home	
	POSIX	GUFI	POSIX	GUFI
Files	**294,188**	**294,188**	**13,360,753**	**13,229,405**
Dirs	13,012	13,012	1,633,564	1,622,424
Time	32.1	0.47	2,040	39.2
Files/sec	9,164	625,931	6,549	337,484

Improved Query Results

	Find all files in NFS Home as uid 12345		Find all files in NFS Home	
	POSIX	GUFI	POSIX	GUFI
Files	294,188	294,188	13,360,753	13,229,405
Dirs	13,012	13,012	1,633,564	1,622,424
Time	**32.1**	**0.47**	2,040	39.2
Files/sec	**9,164**	**625,931**	6,549	337,484

68x

Improved Query Results

	Find all files in NFS Home as uid 12345		Find all files in NFS Home	
	POSIX	GUFI	POSIX	GUFI
Files	294,188	294,188	13,360,753	13,229,405
Dirs	13,012	13,012	1,633,564	1,622,424
Time	32.1	0.47	**2,040**	**39.2**
Files/sec	9,164	625,931	6,549	337,484
		68x		**51x**

Improved Query Results

	Find all files in scratch1 as uid 67890		Find all files in lustre scratch1		Find all files in scratch1 and NFS home as uid 67890	
	POSIX	GUFI	POSIX	GUFI	POSIX	GUFI
Files	**22,771,329**	**22,509,652**	**119,296,067**	**118,509,899**	-	22,522,140
Dirs	240,736	237,759	5,541,230	5,523,153	-	239,603
Time (s)	531.6	14.5	11,309	134.2	-	14.9
Files/s	42,835	1,553,956	10,548	883,413	-	1,511,553

Improved Query Results

	Find all files in scratch1 as uid 67890		Find all files in lustre scratch1		Find all files in scratch1 and NFS home as uid 67890	
	POSIX	GUFI	POSIX	GUFI	POSIX	GUFI
Files	22,771,329	22,509,652	119,296,067	118,509,899	-	22,522,140
Dirs	240,736	237,759	5,541,230	5,523,153	-	239,603
Time (s)	**531.6**	**14.5**	11,309	134.2	-	14.9
Files/s	**42,835**	**1,553,956**	10,548	883,413	-	1,511,553

36x

Improved Query Results

	Find all files in scratch1 as uid 67890		Find all files in lustre scratch1		Find all files in scratch1 and NFS home as uid 67890	
	POSIX	GUFI	POSIX	GUFI	POSIX	GUFI
Files	22,771,329	22,509,652	119,296,067	118,509,899	-	22,522,140
Dirs	240,736	237,759	5,541,230	5,523,153	-	239,603
Time (s)	531.6	14.5	**11,309**	**134.2**	-	14.9
Files/s	42,835	1,553,956	**10,548**	**883,413**	-	1,511,553
		36x		**84x**		

Improved Query Results

	Find all files in scratch1 as uid 67890		Find all files in lustre scratch1		Find all files in scratch1 and NFS home as uid 67890	
	POSIX	GUFI	POSIX	GUFI	POSIX	GUFI
Files	22,771,329	22,509,652	119,296,067	118,509,899	-	22,522,140
Dirs	240,736	237,759	5,541,230	5,523,153	-	239,603
Time (s)	531.6	14.5	11,309	134.2	-	**14.9**
Files/s	42,835	1,553,956	10,548	883,413	-	**1,511,553**
		36x		**84x**		

Index Creation Results

- 118,509,899 files from lustre filesystem into GUFI tree: 148.9s
  - 795,902 files/s
- 13,229,405 files from NFS home filesystem into GUFI tree: 38.4s
  - 344,515 files/s

Next Steps in Performance

- Sharding or scaling-out – at-least testing, in case we need to
  - Serialization in the kernel? dcache?
- Summary tables
- Exploring different file systems – user space
  - Make use of those SSDs

What’s ahead?

Work in Progress – Busy Summer… ☺

- Hardening – code and deployment strategies
- Ingest tools
- Testing other underlying file systems
- Further testing scale-out nature
- Web server and visualization tools

Questions?

- Thank you!
- Test, contribute, file bugs: https://github.com/mar-file-system/GUFI
- Other on-going work and research can be found via Ultrascale Systems Research Center webpage: https://usrc.lanl.gov/
- Join us in our effort to obtain higher efficiency with the Efficient Mission Centric Computing Consortium: https://usrc.lanl.gov/emc3.php